Sarcasm Detection on SARC

Siddhi Visaria
210815867
Julian Hough
M.Sc. Artificial Intelligence

***Abstract*—We are using the Self–Annotated Reddit Corpus (SARC) to detect and analyze the different types of sarcasm and their impact on the community. This is research to understand how sentiment analysis in NLP works when the sentiment is sarcasm**

Keywords—sarcasm, detection, analysis, SARC, sentiment

I. Introduction

This is research to understand how sentiment analysis in Natural Language Processing plays an important role. Sentiments are a way of expressing oneself or one's state of mind. The basic six emotions – happy, sad, anger, disgust, surprise and fear are the ones that can be detected and analysed by sentiment analysis (Ekman, 1972) while we have emotions like sarcasm, laughter, anxiety, joy and many more which are not easily identifiable and comprehendible until the content and context of a text is known. Purver and Battersby (2012), have experimented with distant supervision on emotion classification. For classifying we need corpora, many datasets are available online, and more frequently the source is social media. Social media has a short length of texts with massive amounts of data, Twitter has been a constant source for datasets.

We are going to be detecting sarcasm using Natural Language Processing, which is a complex task for a human brain to understand because of its rare usage and understanding of sarcasm. Oxford Learner's Dictionary defines sarcasm as a way of using words that are the opposite of what you mean to be unpleasant to somebody or to make fun of them.

II. Related Work

Datasets have been built for a very long time, some of them are balanced datasets with a nearly equal number of statements of each label (Gonz´alez-Ib´anez et al., 2011; Bamman and Smith, 2015; Joshi et al., 2015; Amir et al., 2016; Oraby et al., 2016). Riloff et al (2013); Swanson et al (2014); Wallace et al.(2015) have created datasets annotated by a human. Khodak et al (2018) have prepared a corpus of 1.3 million statements which is many times any of the existing datasets, it is known as Self-Annotated Reddit Corpus (SARC). The most frequent source of data is Twitter, for its properties of having short texts, this can lead to balanced or unbalanced, annotated or unannotated, with or without hashtags (Reyes et al (2013); Bamman and Smith (2015); Joshi et al (2015)), emoticons (Read, 2005) and correct punctuations dataset.

Automatic Sarcasm Detection has been of great interest in the field of research for sentiment classification and analysis. For sarcasm detection, we use lexical and pragmatic parameters to identify sarcasm in the statement. Numerous studies have been conducted in this area, including one by Tepperman et al. (2006) which employs prosodic and spectral cues for spoken conversation systems. Carvalho et al. employ linguistic elements like positive predicates, interjections, and gestural indications like emoticons, quotation marks, etc (2009). Davidov et al. (2010), Tsur et al. (2010), and Gonz'alez-Ib'anez et al. (2011), respectively, using syntactic patterns and emoticons. According to Riloff et al. (2013), sarcasm contrasts positively with sentimental language and adverse circumstances. Multiple characteristics, including lexical, pragmatic, implicit, and explicit context incongruity, are used by Joshi et al. (2015).

III. Methodology

A. Dataset

We are using the SARC – Self Annotated Reddit Corpus as our dataset. This dataset is a massive dataset containing both unbalanced labels and self-annotated labels. (Khodak et al, 2018) Reddit is a social networking site where users interact by leaving comments for responses, which are labelled posts made up of text, external links, and/or embedded material and placed on subreddits, or topic-specific forums; examples of subreddits include funny, photos, and science. Users leave comments on submissions and other comments, creating a conversational tree where each remark has a parent comment. Any node in a Reddit link's tree that we refer to as an element (i.e., comments or submissions).

The dataset has the following columns:

- label –0 is not sarcastic and 1 is sarcastic;
- comment – contains comment on parent_comment;
- parent_comment – it is the main comment;
- author – the person who has posted the parent_comment;
- subreddit – the topic of the comments
- scores – ranking of the posted comment
- ups, downs – votes for the comment
- dated, created_utc – stores the date

This dataset is many times the existing dataset.

B. Pre-Processing

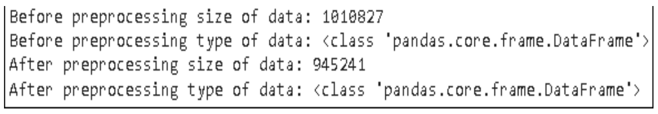
- To use the dataset, we need to pre-process it, which means cleaning the data. Preparing the dataset to use as input for the model.
- Read the file using pandas and then we drop the columns, which will not be needed further and then

check if any of the columns in the data frame are null or not.

- The next step is to check the max length of the comment and parent_comment, resulting in new columns in the data frame, which are then dropped.
- Now delete rows for the label column where the value is not an integer, and for comment and parent_comment when the value is not a string. (See Fig. 1)
- Convert the string to lowercase and remove white spaces, contractions, new lines, and double spaces replaced with a single space and add a white space before and after punctuations.
- On applying these pre-processing steps we observe the length of comment and parent_comment have varied. (See Table 1)
- After all the steps of pre-processing we save our data to a new CSV file, which will be further used to train the models.

Table 1: Length of columns before and after preprocessing

Length	Before preprocessing	After preprocessing
comment	10000	10004
parent_comment	32759	23735



```
Before preprocessing size of data: 1010827
Before preprocessing type of data: <class 'pandas.core.frame.DataFrame'>
After preprocessing size of data: 945241
After preprocessing type of data: <class 'pandas.core.frame.DataFrame'>
```

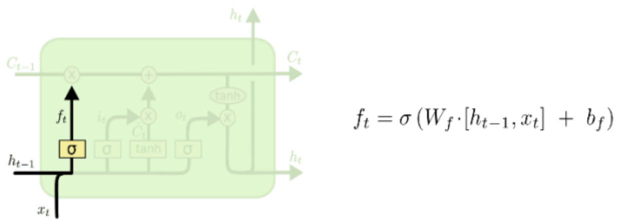
Fig. 1: Size of data before and after removing rows.

C. *Long Short Term Memory (LSTM)*

Recurrent Neural Network (RNN) have short-term memory problems, we have a special type of RNN called "LSTMs" that is capable of learning long-term dependencies. LSTMs are created to prevent long-term reliance issues. They don't stress about learning rather, remembering knowledge for extended periods is their default habit. The LSTM may modify the cell state by removing or adding information, which is carefully controlled via gates.

Information can pass via gates on a purely voluntary basis. They are constructed from a sigmoid neural net layer and a pointwise multiplication operation.

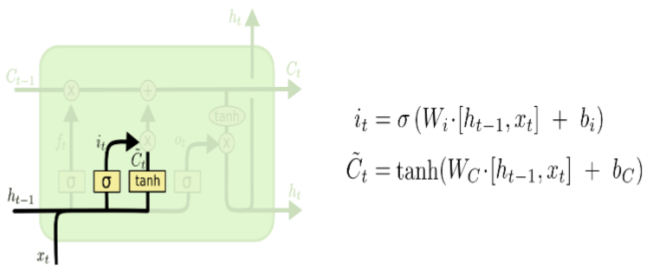
Choosing what information from the cell state to discard is the first stage in our LSTM. The "forget gate layer," a sigmoid layer, decides on this. It examines h_{t-1} and x_t, and for each number in the cell state C_{t-1}, it produces a number between 0 and 1. A 1 means "entirely keep this," whereas 0 means "entirely get rid of this." (See Fig. 2)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Fig. 2: Forget Gate Layer[1]

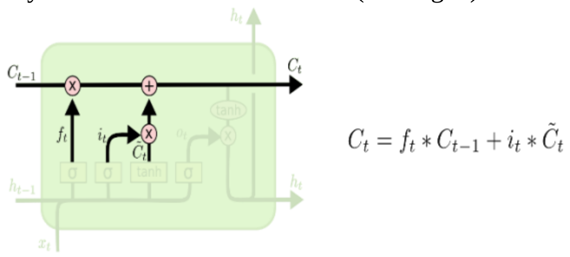
The next step is to choose the new data that will be kept in the cell state. This comprises of two parts: the "input gate layer," a sigmoid layer, first determines which values will be updated. The state is then updated with a vector of potential new values, C_t, created by a tanh layer. These two will be combined in the subsequent phase to provide an update on the state. (See Fig. 3)



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Fig. 3 Input Gate Layer[1]

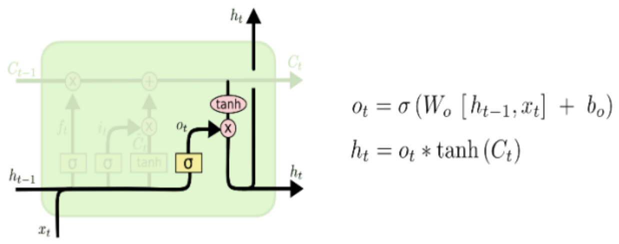
Now update the old cell state, C_{t-1}, into the new cell state C_t. We just need to perform what is already decided.
The old state is multiplied by f_t, forgetting the things that are already decided. Then we add $i_t * C_t$. (See Fig. 4)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fig. 4 Update Gate Layer[1]

Now it is decided what is going to be the output, run sigmoid layer to decide what part of the cell state is passed to the output. Pass the cell state through tanh and multiply with the sigmoid gate's output. Finally, we get only the parts that were already decided. (See Fig. 5)



$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Fig. 5 Output Gate Layer[1]

We have used Bidirectional LSTM (BiLSTM) on our dataset. In BiLSTM the input flows in both directions, and it's capable of utilizing information from both sides. (See Fig. 6)

[1] Christopher Olah (2015) Understanding LSTM Networks Available from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ [online]

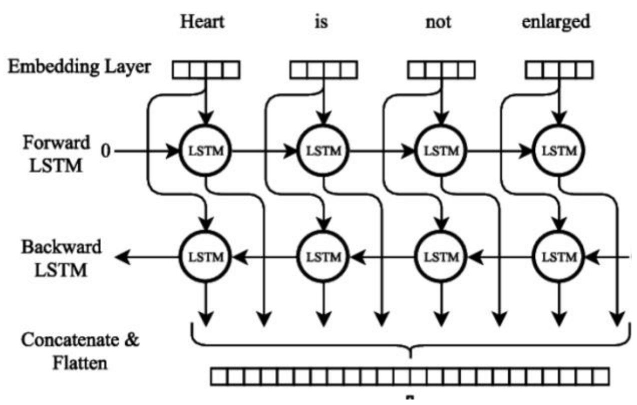
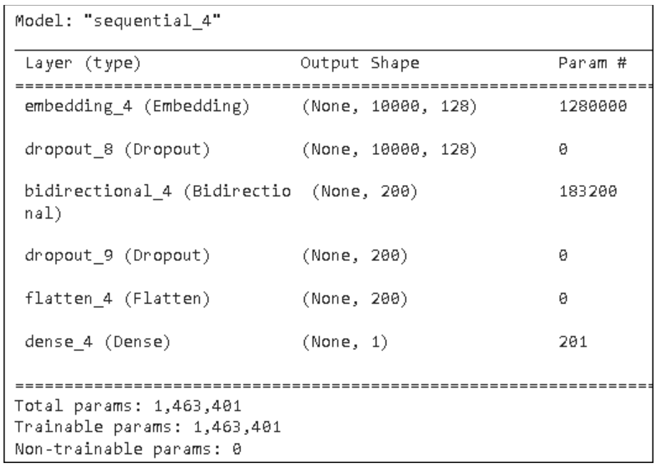


Fig. 6 Architecture of Bidirectional LSTM (BiLSTM)



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 10000, 128)        1280000

 dropout_8 (Dropout)         (None, 10000, 128)        0

 bidirectional_4 (Bidirectio  (None, 200)              183200
 nal)

 dropout_9 (Dropout)         (None, 200)               0

 flatten_4 (Flatten)         (None, 200)               0

 dense_4 (Dense)             (None, 1)                 201

=================================================================
Total params: 1,463,401
Trainable params: 1,463,401
Non-trainable params: 0
```

Fig. 7 Bidirectional LSTM structure built for our problem.

D. *Convolutional Neural Network (CNN)*

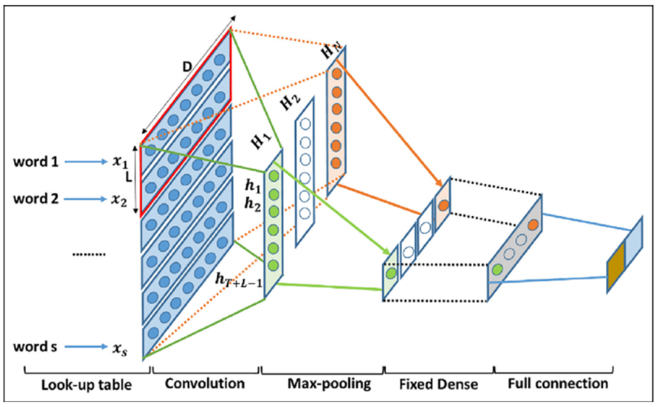


Fig. 8 Architecture of Convolutional Neural Network (CNN)

Convolution is the process of combining two connections mathematically to create a third relationship, bringing together two sets of information. Convolution over input: Using a filter or kernel, we apply convolution to the input data to extract the features. When extracting features, this is crucial. The sliding filter has certain options, such as how much data to accept at once and how much input should be overlapped. Stride is the size of the step when a filter moves at every time step. Filter Count is the number of filters to be used in the network.

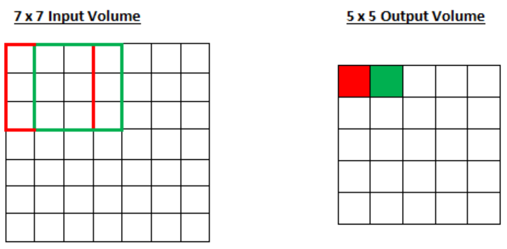


Fig. 9 Stride in Convolutional Neural Network (CNN)

An activation function is passed on the output once the feature maps have been generated, so the output will be non-linear. To avoid shrinking of feature map padding is done around the input. To avoid overfitting the model, a pooling layer is used between the convolutional layers, so that the dimensional complexity reduces. (See Fig. 8,9,10)

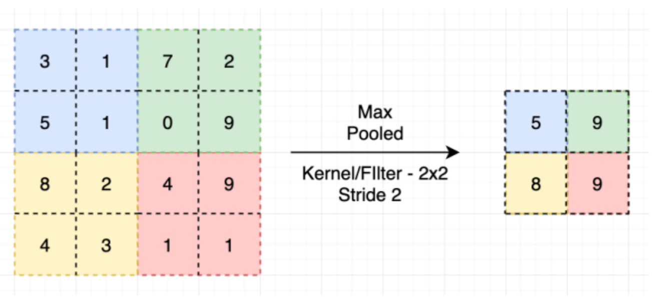
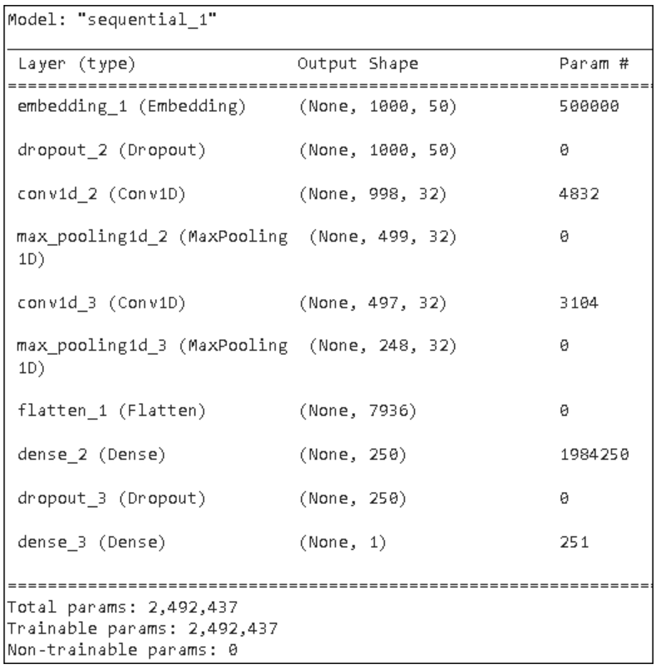


Fig.10: Pooling in Convolutional Neural Network (CNN)



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 1000, 50)          500000

 dropout_2 (Dropout)         (None, 1000, 50)          0

 conv1d_2 (Conv1D)           (None, 998, 32)           4832

 max_pooling1d_2 (MaxPooling  (None, 499, 32)          0
 1D)

 conv1d_3 (Conv1D)           (None, 497, 32)           3104

 max_pooling1d_3 (MaxPooling  (None, 248, 32)          0
 1D)

 flatten_1 (Flatten)         (None, 7936)              0

 dense_2 (Dense)             (None, 250)               1984250

 dropout_3 (Dropout)         (None, 250)               0

 dense_3 (Dense)             (None, 1)                 251

=================================================================
Total params: 2,492,437
Trainable params: 2,492,437
Non-trainable params: 0
```

Fig. 11 CNN structure built for our problem

IV. Results

We are training the Reddit dataset using neural network models such as BiLSTM and CNN. Three separate inputs were used in each model: comment, parent_comment, and concatenation column (“comment” and “parent_comment”).

A. *Experiment Results*

Table 2. Loss incurred for BiLSTM and CNN during Training

Loss	comment	parent_comment	comment<SEP> parent_comment
BiLSTM	16.81%	28.42%	20.95%
CNN	38.58%	40.80%	32.51%

Table 3. The accuracy achieved for BiLSTM and CNN during Training

accuracy	comment	parent_comment	comment<SEP> parent_comment
BiLSTM	92.95%	87.38%	91.69%
CNN	80.97%	79.38%	85.87%

As per Tables 2 and 3, we understand that the accuracy for BiLSTM is higher than that for CNN, which means that the BiLSTM model has performed better than CNN. Let's see in the next section how each model performed prediction on sentences.

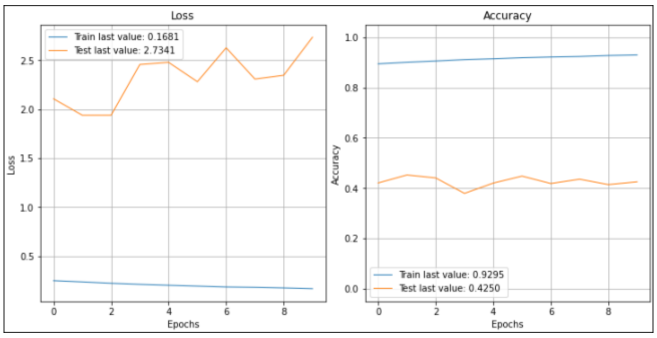


Fig.12: Loss and Accuracy graph for BiLSTM for ‘comment’

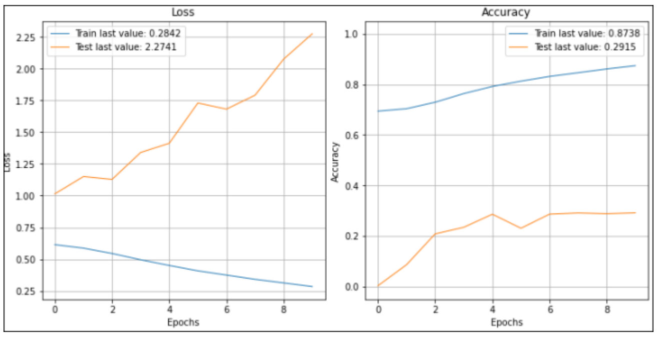


Fig.13: Loss and Accuracy graph for BiLSTM for ‘parent_comment’

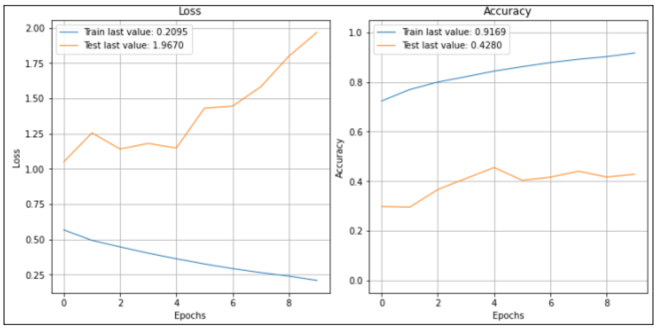


Fig.14: Loss and Accuracy graph for BiLSTM for ‘comment<SEP>parent_comment’

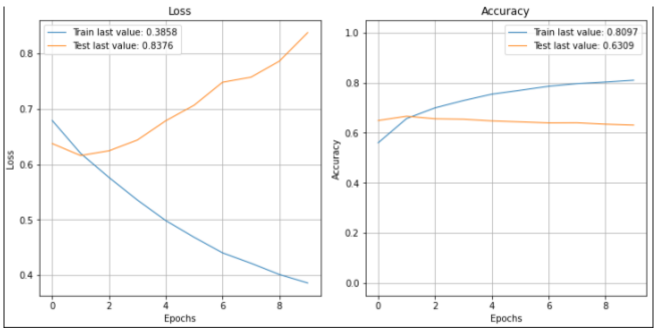


Fig.15: Loss and Accuracy graph for CNN for ‘comment’

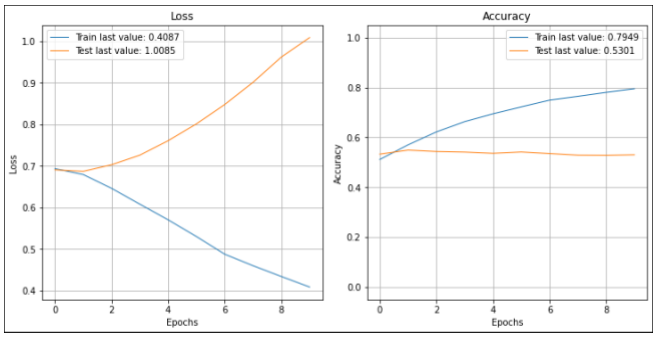


Fig.16: Loss and Accuracy graph for CNN for ‘parent_comment’

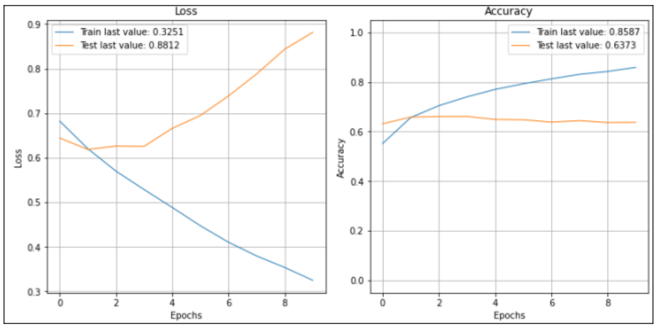


Fig.17: Loss and Accuracy graph for CNN for ‘comment<SEP>parent_comment’

B. *Evaluation and Validation*

Table 4. Probability of statement being sarcastic for **BiLSTM**

types of sents		lstm	lstm_parent	concat_lstm
universal		12.12%	23.69%	2.80%
non sarcastic	comment	14.13%	22.19%	2.76%
	parent_comment	14.24%	21.34%	3.06%
sarcastic	comment	10.18%	24.36%	2.89%
	parent_comment	13.04%	21.22%	2.88%

Table 5. Probability of the statement being sarcastic for **CNN**

types of sents		cnn	cnn_parent	concat_cnn
universal		31.42%	43.45%	26.13%
non sarcastic	comment	41.83%	37.95%	27.94%
	parent_comment	88.54%	8.93%	68.57%
sarcastic	comment	93.12%	56.29%	67.23%
	parent_comment	6.33%	88.43%	19.08%

Table 6. Loss incurred for BiLSTM & CNN during Evaluating

loss	comment	parent_comment	comment<SEP> parent_comment
BiLSTM	26.21%	22.29%	19.13%
CNN	83.75%	10.08%	88.12%

Table 7. The accuracy achieved for BiLSTM & CNN during Evaluating

accuracy	comment	parent_comment	comment<SEP> parent_comment
BiLSTM	43.02%	29.98%	44.11%
CNN	63.09%	53.00%	63.73%

Conclusion

From Tables 2 and 3, we can conclude that the training accuracy for the Bidirectional LSTM model is higher in all three cases as compared to the CNN model.

When validating the trained models on the same sentences for Bidirectional LSTM and CNN we observe inverse results from that of training accuracy. The CNN model predicts the sarcastic 'comment' correctly while the sarcastic 'parent_comment' is partially correct. The non-sarcastic 'comment' are predicted correctly and the non-sarcastic 'parent_comment' are predicted partially correct by the CNN model. Though the CNN model has accuracy lower than BiLSTM the predictions for the statements are better.

The BiLSTM predicts the non-sarcastic and universal statements correctly for all three cases.

On evaluation, we notice that the validation loss and validation accuracy for CNN is too higher than BiLSTM.

To conclude, we can say that the data prediction for the statements has not been that accurate for the "sarcastic" sentiments for both the parent_comment and comment columns value (See Table 4). We can see that wrong values have been predicted which indicates the sarcastic statement to be non-sarcastic.

According to me, the statement labelling in the dataset is not correct. For example, I feel the statements that have been tested for non-sarcastic are sarcastic and vice-versa.

From our experiments, we conclude that sarcasm is about one's view or thinking, basically all about individual mindset. The authors (Khodak et al, 2018) of the SARC dataset might have felt a particular comment to be sarcastic while the other might not be, hence resulting in a difference in labelling.

Future Work

In future, the existing models and dataset need to be modified and worked upon to get the right predictions, concerning the other columns of the dataset.

References

Aditya Joshi, Pushpak Bhattacharyya, and Mark J. Carman. 2017. Automatic Sarcasm Detection: A Survey. ACM Comput. Surv. 50, 5, Article 73 (September 2017), 22 pages. https://doi.org/10.1145/3124420

Amir, S., Wallace, B. C., Lyu, H., Carvalho, P., and Silva, M. J. (2016). Modelling context with user embeddings for sarcasm detection in social media.

Bamman, D. and Smith, N. A. (2015). Contextualized sarcasm detection on Twitter. Association for the Advancement of Artificial Intelligence.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences on Twitter and amazon. In Proceedings of the fourteenth conference on computational natural language learning, pages 107–116. Association for Computational Linguistics.

Gonz´alez-Ib´anez, R., Muresan, S., and Wacholder, N. (2011). Identifying sarcasm in Twitter: A closer look. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 581–586. Association for Computational Linguistics.

Jonathon Read. 2005. Using emoticons to reduce dependency on machine learning techniques for sentiment classification. In Proceedings of the 43rd Meeting of the Association for Computational Linguistics.

Joseph Tepperman, David Traum, and Shrikanth Narayanan. 2006. "yeah right": Sarcasm recognition for spoken dialogue systems. In Ninth International Conference on Spoken Language Processing.

Joshi, A., Sharma, V., and Bhattacharyya, P. (2015). Harnessing context incongruity for sarcasm detection. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics pages 757–762. Association for Computational Linguistics.

Matthew Purver and Stuart Battersby, Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 482–491, Avignon, France, April 23 - 27 2012. c 2012 Association for Computational Linguistics.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. arXiv preprint arXiv:1704.05579.

Oraby, S., Harrison, V., Reed, L., Hernandez, E., Riloff, E., and Walker, M. (2016). Creating and characterizing a diverse corpus of sarcasm in dialogue. In Proceedings of the SIGDIAL 2016 Conference, pages 31–41. Association for Computational Linguistics.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In ICWSM, pages 162–169.

Paula Carvalho, Lu´ıs Sarmento, M´ario J Silva, and Eug´enio De Oliveira. 2009. Clues for detecting irony in user-generated content: oh...!! it’s so easy;-. In Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion, pages 53–56. ACM.

Roberto Gonz´alez-Ib´anez, Smaranda Muresan, and NinaWacholder. 2011. Identifying sarcasm in twitter: a closer look. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2, pages 581–586. Association for Computational Linguistics.

Reyes, A., Rosso, P., and Veale, T. (2013). A multidimensional approach for detecting irony in Twitter. Data Knowledge Engineering, 47(1).

Riloff, E., Qadir, A., Surve, P., Silva, L. D., Gilbert, N., and Huang, R. (2013). Sarcasm is the contrast between a positive sentiment and a negative situation. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 704–741. Association for Computational Linguistics.

Swanson, R., Lukin, S., Eisenberg, L., Corcoran, T. C., and Walker, M. A. (2014). Getting reliable annotations for sarcasm in online dialogues. In Language Resources and Evaluation Conference.

Wallace, B. C., Choe, D. K., and Charniak, E. (2015). Sparse, contextually informed models for irony detection: Exploiting user communities, entities, and sentiment. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, pages 1035–1044. Association for Computational Linguistics.